

Linguistic Data Management

Steven Bird

University of Melbourne, AUSTRALIA

August 27, 2008

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
 - data types: texts and lexicons
 - useful ways to access data using Python: csv, html, xml
 - adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Introduction

- language resources, types, proliferation
- role in NLP, CL
- enablers: storage/XML/Unicode; digital publication; resource catalogues
- obstacles: discovery, access, format, tool
- data types: texts and lexicons
- useful ways to access data using Python: csv, html, xml
- adding a corpus to NLTK

Linguistic Databases

- Field linguistics
- Corpora
- Reference Corpus

Linguistic Databases

- Field linguistics
- Corpora
- Reference Corpus

Linguistic Databases

- Field linguistics
- Corpora
- Reference Corpus

Fundamental Data Types

Lexicon

Abstraction: fielded records

key	field	field	field	field
key	field	field	field	field

Eg: dictionary

wake: werk, [v], cease to sleep...
walk: wo:k, [v], progress by lifting and setting down each foot...

Eg: comparative wordlist

wake; aufwecken; acordar
walk; gehen; andar
write; schreiben; enscrever

Eg: verb paradigm

wake woke woken
write wrote written
wring wrung wrung

Text

Abstraction: time series



Eg: written text

A long time ago, Sun and Moon lived together. They were good brothers. ...

Eg: POS-tagged text

A/DT long/JJ time/NN ago/RB .,
Sun/NNP and/CC Moon/NNP
lived/VBD together/RB .,

Eg: interlinear text

Ragaipa irai vateri
ragai -pa ira -i vate -ri
PP.1.SG -BEN RP.3.SG.M -ABS give -2.SG

Example: TIMIT

- TI (Texas Instruments) + MIT
 - balance
 - sentence selection
 - layers of annotation
 - speaker demographics, lexicon
 - combination of time-series and record-structured data
 - programs for speech corpus

Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

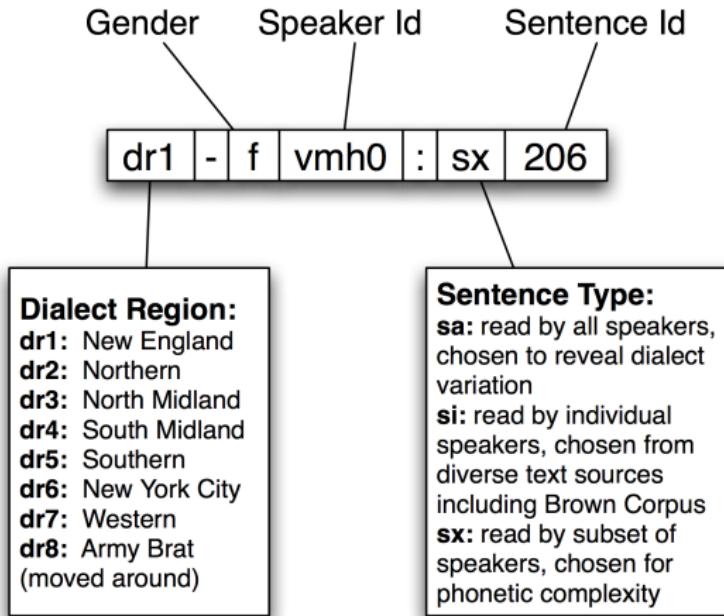
Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

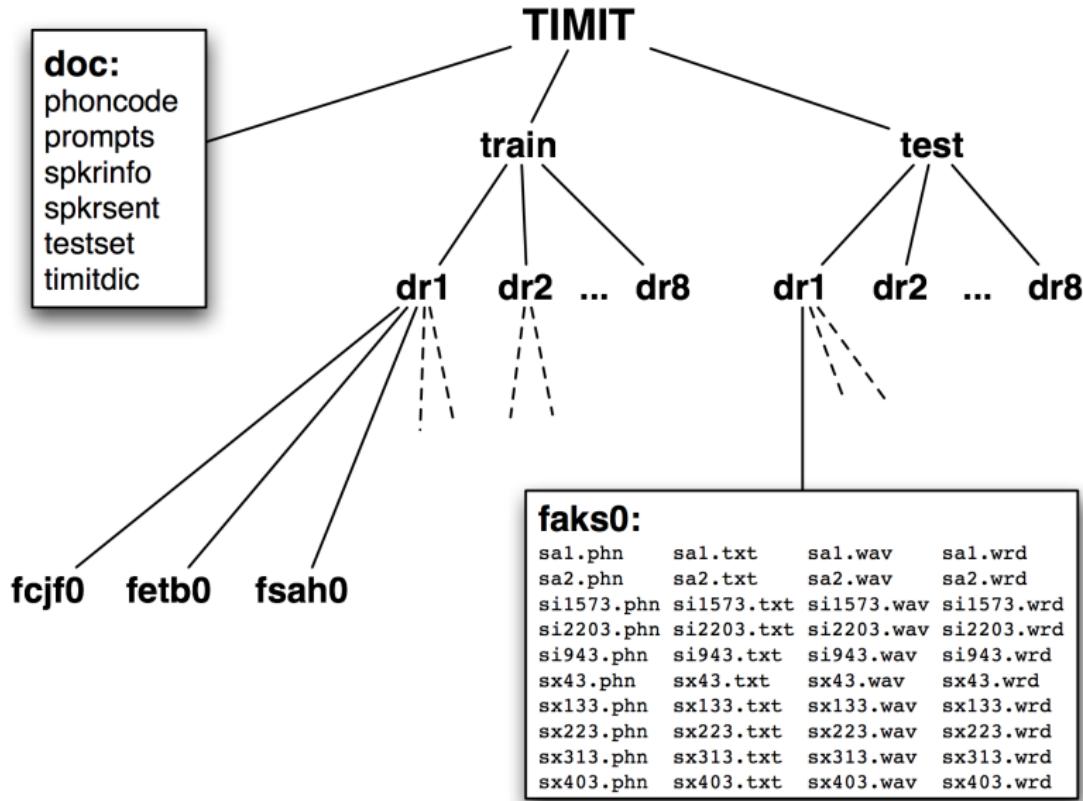
Example: TIMIT

- TI (Texas Instruments) + MIT
- balance
- sentence selection
- layers of annotation
- speaker demographics, lexicon
- combination of time-series and record-structured data
- programs for speech corpus

Example: TIMIT



Example: TIMIT



Example: TIMIT

```
>>> phonetic = nltk.corpus.timit.phones('dr1-fvmh0/sal')
>>> phonetic
['h#', 'sh', 'iy', 'hv', 'ae', 'dcl', 'y', 'ix', 'dcl', 'd', 'aa',
's', 'ux', 'tcl', 'en', 'gcl', 'g', 'r', 'iy', 's', 'iy', 'w', 'aa',
'sh', 'epi', 'w', 'aa', 'dx', 'ax', 'q', 'ao', 'l', 'y', 'ih', 'ax'
>>> nltk.corpus.timit.word_times('dr1-fvmh0/sal')
[('she', 7812, 10610), ('had', 10610, 14496), ('your', 14496, 1579,
('dark', 15791, 20720), ('suit', 20720, 25647), ('in', 25647, 2690,
('greasy', 26906, 32668), ('wash', 32668, 37890), ('water', 38531,
('all', 43091, 46052), ('year', 46052, 50522)]
```

Example: TIMIT

```
>>> timitdict = nltk.corpus.timit.transcription_dict()
>>> timitdict['greasy'] + timitdict['wash'] + timitdict['water']
['g', 'r', 'iy1', 's', 'iy', 'w', 'aol', 'sh', 'w', 'aol', 't', 'a
>>> phonetic[17:30]
['g', 'r', 'iy', 's', 'iy', 'w', 'aa', 'sh', 'epi', 'w', 'aa', 'dx
>>> nltk.corpus.timit.spkrinfo('dr1-fvmh0')
SpeakerInfo(id='VMH0', sex='F', dr='1', use='TRN', recdate='03/11/
birthdate='01/08/60', ht='5\'05"', race='WHT', edu='BS',
comments='BEST NEW ENGLAND ACCENT SO FAR')
```

Lifecycle

- **create**
- annotate texts
- refine lexicon
- organize structure
- publish

Lifecycle

- create
- annotate texts
- refine lexicon
- organize structure
- publish

Lifecycle

- create
- annotate texts
- refine lexicon
- organize structure
- publish

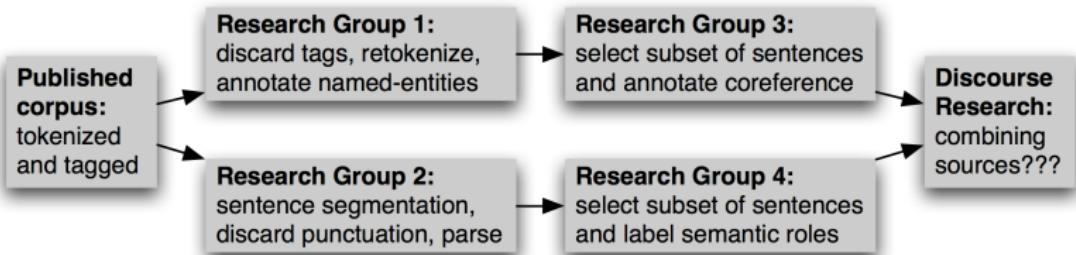
Lifecycle

- create
- annotate texts
- refine lexicon
- organize structure
- publish

Lifecycle

- create
- annotate texts
- refine lexicon
- organize structure
- publish

Evolution



Creating Data: Primary Data

- spiders
- recording
- texts

Creating Data: Primary Data

- spiders
- recording
- texts

Creating Data: Primary Data

- spiders
- recording
- texts

Data Cleansing: Accessing Spreadsheets

```
dict.csv:  
"sleep","sli:p","v.i","a condition of body and mind ..."  
"walk","wo:k","v.intr","progress by lifting and setting down each foot  
"wake","weik","intrans","cease to sleep"  
  
>>> import csv  
>>> file = open("dict.csv", "rb")  
>>> for row in csv.reader(file):  
...     print row  
['sleep', 'sli:p', 'v.i', 'a condition of body and mind ...']  
['walk', 'wo:k', 'v.intr', 'progress by lifting and setting down e  
['wake', 'weik', 'intrans', 'cease to sleep']
```

Data Cleansing: Validation

```
def undefined_words(csv_file):
    import csv
    lexemes = set()
    defn_words = set()
    for row in csv.reader(open(csv_file)):
        lexeme, pron, pos, defn = row
        lexemes.add(lexeme)
        defn_words.union(defn.split())
    return sorted(defn_words.difference(lexemes))

>>> print undefined_words("dict.csv")
['...', 'a', 'and', 'body', 'by', 'cease',
 'condition', 'down', 'each', 'foot',
 'lifting', 'mind', 'of', 'progress',
 'setting', 'to']
```

Data Cleansing: Accessing Web Text

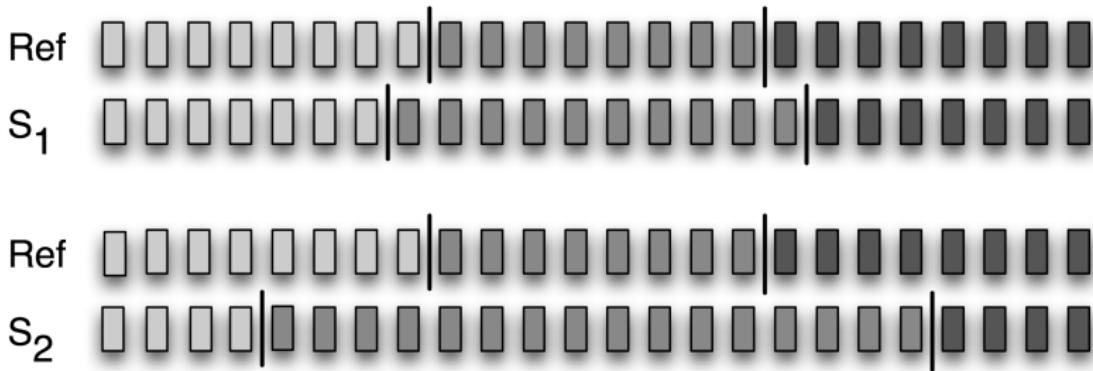
```
>>> import urllib, nltk  
>>> html = urllib.urlopen('http://en.wikipedia.org/').read()  
>>> text = nltk.clean_html(html)  
>>> text.split()  
['Wikimedia', 'Error', 'WIKIMEDIA', 'FOUNDATION', 'Fout', 'Fel',  
'Fallo', '\xe9\x94\x99\x8e\xaf\xaf', '\xe9\x8c\xaf\xe8\xaa\xaa',  
'Erreur', 'Error', 'Fehler', '\xe3\x82\x8e\x83\x83\x9e\x83\xbc',  
'B\xc5\x82\xc4\x85d', 'Errore', 'Erro', 'Chyba', 'EnglishThe',  
'Wikimedia', 'Foundation', 'servers', 'are', 'currently',  
'experiencing', 'technical', 'difficulties.The', 'problem', 'is',  
'most', 'likely', 'temporary', 'and', 'will', 'hopefully', 'be',  
'fixed', 'soon.', 'Please', 'check', 'back', 'in', 'a', 'few',  
'minutes.For', 'further', 'information,', 'you', 'can', 'visit',  
'the', 'wikipedia', 'channel', 'on', 'the', 'Freenode', 'IRC', ...
```

Creating Data: Annotation

- linguistic annotation
- Tools: <http://www.exmaralda.org/annotation/>

Creating Data: Inter-Annotator Agreement

- Kappa statistic
- Windowdiff



Processing Toolbox Data

- single most popular tool for managing linguistic field data
- many kinds of validation and formatting not supported by Toolbox software
- each file is a collection of *entries* (aka *records*)
- each entry is made up of one or more *fields*
- we can apply our programming methods, including chunking and parsing

Processing Toolbox Data

- single most popular tool for managing linguistic field data
- many kinds of validation and formatting not supported by Toolbox software
- each file is a collection of *entries* (aka *records*)
- each entry is made up of one or more *fields*
- we can apply our programming methods, including chunking and parsing

Processing Toolbox Data

- single most popular tool for managing linguistic field data
- many kinds of validation and formatting not supported by Toolbox software
- each file is a collection of *entries* (aka *records*)
- each entry is made up of one or more *fields*
- we can apply our programming methods, including chunking and parsing

Processing Toolbox Data

- single most popular tool for managing linguistic field data
- many kinds of validation and formatting not supported by Toolbox software
- each file is a collection of *entries* (aka *records*)
- each entry is made up of one or more *fields*
- we can apply our programming methods, including chunking and parsing

Processing Toolbox Data

- single most popular tool for managing linguistic field data
- many kinds of validation and formatting not supported by Toolbox software
- each file is a collection of *entries* (aka *records*)
- each entry is made up of one or more *fields*
- we can apply our programming methods, including chunking and parsing

Toolbox Example

```
\lx kaa
\ps N.M
\cl isi
\ge cooking banana
\gp banana bilong kukim
\sf FLORA
\dt 12/Feb/2005
\ex Taeavi iria kaa isi kovopaueva kaparapasia.
\xp Taeavi i bin planim gaden banana bilong kukim tasol
\xe Taeavi planted banana in order to cook it.
```

Accessing Toolbox Data

- scan the file, convert into tree object
- preserves order of fields, gives array and XPath-style access

```
>>> from nltk.corpus import toolbox  
>>> lexicon = toolbox.xml('rotokas.dic')
```

Accessing with Indexes

```
>>> lexicon[3][0]
<Element lx at 77bd28>
>>> lexicon[3][0].tag
'lx'
>>> lexicon[3][0].text
'kaa'
```

Accessing with Indexes (cont)

```
>>> print nltk.corpus.reader.toolbox.to_sfm_string(lexicon[3])
\lx kaa
\ps N.M
\cl isi
\ge cooking banana
\gp banana bilong kukim
\sf FLORA
\dt 12/Feb/2005
\ex Taeavi iria kaa isi kovopaueva kaparapasia.
\xp Taeavi i bin planim gaden banana bilong kukim tasol long paia.
\xe Taeavi planted banana in order to cook it.
```

Accessing with Paths

```
>>> [lexeme.text.lower() for lexeme in lexicon.findall('record/lx')]  
['kaa', 'kaa', 'kaa', 'kaakaaro', 'kaakaaviko', 'kaakaavo', 'kaakaoko',  
'kaakasi', 'kaakau', 'kaakauko', 'kaakito', 'kaakuupato', ..., 'kuvuto']
```

- lexicon is a series of record objects
- each contains field objects, such as lx and ps
- address all the lexemes: record/lx

Data Cleansing: Toolbox

- parsing (Listing 4)
- chunking (Listing 5)
- adding missing fields (next)

Data Cleansing: Toolbox

- parsing (Listing 4)
- chunking (Listing 5)
- adding missing fields (next)

Data Cleansing: Toolbox

- parsing (Listing 4)
- chunking (Listing 5)
- adding missing fields (next)

Adding New Fields

- Example: add CV field
- Aside: utility function to do CV template

```
>>> import re
>>> def cv(s):
...     s = s.lower()
...     s = re.sub(r'[^a-z]',      r'_', s)
...     s = re.sub(r'[aeiou]',    r'V', s)
...     s = re.sub(r'^V_]',      r'C', s)
...     return (s)
```

Adding New Fields (cont)

```
>>> from nltk.etree.ElementTree import SubElement  
>>> for entry in lexicon:  
...     for field in entry:  
...         if field.tag == 'lx':  
...             cv_field = SubElement(entry, 'cv')  
...             cv_field.text = cv(field.text)
```

Adding New Fields (cont)

```
>>> toolbox.to_sfm_string(lexicon[50])
\lx kaeviro
\cv CVVCVVCV
\ps V.A
\ge lift off
\ge take off
\gp go antap
\nt used to describe action of plane
\dt 12/Feb/2005
\ex Pita kaeviroroe kepa kekesia oa vuripierevo kiuvu.
\xp Pita i go antap na lukim haus win i bagarapim.
\xe Peter went to look at the house that the wind destroyed.
```

Generating HTML Tables from Toolbox Data

```
>>> html = "<table>\n"
>>> for entry in lexicon[70:80]:
...     lx = entry.findtext('lx')
...     ps = entry.findtext('ps')
...     ge = entry.findtext('ge')
...     html += "  <tr><td>%s</td><td>%s</td><td>%s</td></tr>\n" %
...           (lx, ps, ge)
>>> html += "</table>"
>>> print html

<table>
  <tr><td>kakapikoto</td><td>N.N2</td><td>newborn baby</td></tr>
  <tr><td>kakapu</td><td>V.B</td><td>place in sling for purpose of carrying</td></tr>
  <tr><td>kakapua</td><td>N.N</td><td>sling for lifting</td></tr>
  <tr><td>kakara</td><td>N.N</td><td>bracelet</td></tr>
  <tr><td>Kakarapaia</td><td>N.PN</td><td>village name</td></tr>
  <tr><td>kakarau</td><td>N.F</td><td>stingray</td></tr>
  <tr><td>Kakarera</td><td>N.PN</td><td>name</td></tr>
  <tr><td>Kakareraia</td><td>N.??</td><td>name</td></tr>
  <tr><td>kakata</td><td>N.F</td><td>cockatoo</td></tr>
  <tr><td>kakate</td><td>N.F</td><td>bamboo tube for water</td></tr>
</table>
```

Generating XML

```
>>> import sys
>>> from nltk.etree.ElementTree import ElementTree
>>> tree = ElementTree(lexicon[3])
>>> tree.write(sys.stdout)

<record>
  <lx>kaakaaro</lx>
  <ps>N.N</ps>
  <ge>mixtures</ge>
  <gp>???</gp>
  <eng>mixtures</eng>
  <eng>charm used to keep married men and women youthful and attractiv
  <cmt>Check vowel length. Is it kaakaaro or kaakaro?</cmt>
  <dt>14/May/2005</dt>
  <ex>Kaakaroto ira purapaiiveira aue iava opita, voeao-pa airepa oraou
  <xp>Kokonas ol i save wokim long ol kain samting bilong ol nupela ma
  <xe>Mixtures are made from coconut, ???.</xe>
</record>
```

Analysis: Reduplication

- create a table of lexemes and their glosses

```
>>> lexgloss = {}
>>> for entry in lexicon:
...     lx = entry.findtext('lx')
...     if lx and entry.findtext('ps')[0] == 'V':
...         lexgloss[lx] = entry.findtext('ge')
```

- For each lexeme, check if the lexicon contains the reduplicated form:

```
>>> for lex in lexgloss:
...     if lex+lex in lexgloss:
...         print "%s (%s); %s (%s)" % (lex, lexgloss[lex], lex+lex,
```

Reduplication (cont)

kuvu (fill.up); kuvukuvu (stamp the ground)
kitu (save); kitukitu (scrub clothes)
kopa (ingest); kopakopa (gulp.down)
kasi (burn); kasikasi (angry)
koi (high pitched sound); koikoi (groan with pain)
kee (chip); keekee (shattered)
kauo (jump); kauokauo (jump up and down)
kea (deceived); keakea (lie)
kove (drop); kovekove (drip repeatedly)
cape (unable to meet); kapekape (grip with arms not meeting)
kapo (fasten.cover.strip); kapokapo (fasten.cover.strips)
koa (skin); koakoa (remove the skin)
kipu (paint); kipukipu (rub.on)
koe (spoon out a solid); koekoe (spoon out)
kovo (work); kovokovo (surround)
kiru (have sore near mouth); kirukiru (crisp)
kotu (bite); kotukotu (grind teeth together)
kavo (collect); kavokavo (work black magic)
...

Complex Search Criteria

```
>>> from nltk import tokenize, FreqDist  
>>> fd = FreqDist()  
>>> lexemes = [lexeme.text.lower() for lexeme in lexicon.findall('reco  
>>> for lex in lexemes:  
...     for syl in tokenize regexp(lex, pattern=r'[^aeiou][aeiou]'):  
...         fd.inc(syl)
```

- for phonological description, identify segments, alternations, syllable canon...
- what syllable types occur in lexemes (MSC, conspiracies)?

Analysis: Complex Search Criteria (cont)

- Tabulate the results:

```
>>> for vowel in 'aeiou':  
...     for cons in 'ptkvsr':  
...         print '%s%s:%4d' %  
...             (cons, vowel, fd.count(cons+vowel)),  
...     print  
pa: 84  ta: 43  ka: 414  va: 87  sa: 0  ra: 185  
pe: 32  te: 8   ke: 139  ve: 25  se: 1  re: 62  
pi: 97  ti: 0   ki: 88   vi: 96  si: 95  ri: 83  
po: 31  to: 140  ko: 403  vo: 42  so: 3  ro: 86  
pu: 49  tu: 35  ku: 169  vu: 44  su: 1  ru: 72
```

- NB t and s columns
- ti not attested, while si is frequent: palatalization?
- which lexeme contains su? *kasuari*

Analysis: Finding Minimal Sets

- E.g. mace vs maze, face vs faze
- minimal set parameters: context, target, display

Minimal Set	Context	Target	Display
<i>bib, bid, big</i>	first two letters	third letter	word
<i>deal (N), deal (V)</i>	whole word	pos	word (pos)

Analysis: Finding Minimal Sets

- E.g. mace vs maze, face vs faze
- minimal set parameters: context, target, display

Minimal Set	Context	Target	Display
<i>bib, bid, big</i>	first two letters	third letter	word
<i>deal (N), deal (V)</i>	whole word	pos	word (pos)

Finding Minimal Sets: Example 1

```
>>> from nltk import MinimalSet
>>> pos = 1
>>> ms = MinimalSet((lex[:pos] + '_' + lex[pos+1:], lex[pos], lex)
...                     for lex in lexemes if len(lex) == 4)
>>> for context in ms.contexts(3):
...     print context + ':',
...     for target in ms.targets():
...         print "%-4s" % ms.display(context, target, "-"),
...     print
k_si: kasi -    kesi -    kosi
k_ru: karu kiru keru kuru koru
k_pu: kapu kipu -    -    kopu
k_ro: karo kiro -    -    koro
k_ri: kari kiri keri kuri kori
k_pa: kapa -    kepa -    kopa
k_ra: kara kira kera -    kora
k_ku: kaku -    -    kuku koku
k_ki: kaki kiki -    -    koki
```

Finding Minimal Sets: Example 2

```
>>> entries = [(e.findtext('lx'), e.findtext('ps'), e.findtext('ge'))
...             for e in lexicon
...             if e.findtext('lx') and e.findtext('ps') and e.findtext('ge'))
>>> ms = MinimalSet((lx, ps[0], "%s (%s)" % (ps[0], ge))
...                   for (lx, ps, ge) in entries)
>>> for context in ms.contexts()[:10]:
...     print "%10s:" % context, ".join(ms.display_all(context))"
kokovara: N (unripe coconut); V (unripe)
kapua: N (sore); V (have sores)
koie: N (pig); V (get pig to eat)
kovo: C (garden); N (garden); V (work)
kavori: N (crayfish); V (collect crayfish or lobster)
korita: N (cutlet?); V (dissect meat)
keru: N (bone); V (harden like bone)
kirokiro: N (bush used for sorcery); V (write)
kaapie: N (hook); V (snag)
kou: C (heap); V (lay egg)
```

Adding a Corpus to NLTK

- corpus path
- corpus readers

Adding a Corpus to NLTK

- corpus path
- corpus readers

Publishing

- metadata: DC, OLAC
- repositories
- search
- demo

Publishing

- metadata: DC, OLAC
- repositories
- search
- demo

Publishing

- metadata: DC, OLAC
- repositories
- search
- demo

Publishing

- metadata: DC, OLAC
- repositories
- search
- demo